MeshPose: Unifying DensePose and 3D Body Mesh reconstruction

Eric-Tuan Lê[1*†] Antonis Kakolyris[2*] Petros Koutras[2] Himmy Tam[2] Efstratios Skordos[2]
George Papandreou[2] Rıza Alp Güler[2] Iasonas Kokkinos[2]

[1]University College London [2]Snap Inc.

meshpose.github.io

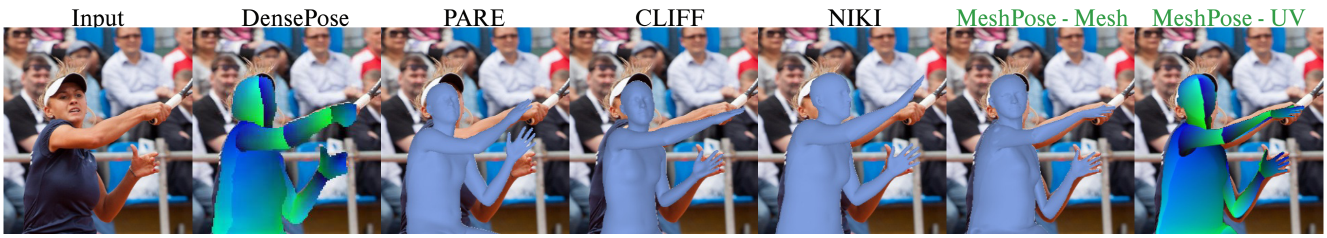


Figure 1. DensePose prediction systems are pixel-accurate but do not provide a 3D mesh, while human mesh recovery systems do not provide pixel-accurate 2D reprojection. We propose MeshPose, a novel human mesh recovery method that combines the benefits of both.

Abstract

DensePose provides a pixel-accurate association of images with 3D mesh coordinates, but does not provide a 3D mesh, while Human Mesh Reconstruction (HMR) systems have high 2D reprojection error, as measured by DensePose localization metrics. In this work we introduce MeshPose to jointly tackle DensePose and HMR. For this we first introduce new losses that allow us to use weak DensePose supervision to accurately localize in 2D a subset of the mesh vertices ('VertexPose'). We then lift these vertices to 3D, yielding a low-poly body mesh ('MeshPose'). Our system is trained in an end-to-end manner and is the first HMR method to attain competitive DensePose accuracy, while also being lightweight and amenable to efficient inference, making it suitable for real-time AR applications.

1. Introduction

3D Human Mesh Reconstruction (HMR) has received increased attention thanks to its broad AR/VR applications such as human-computer interaction, motion capture, entertainment/VFX and virtual try-on. Despite rapid progress in HMR, mesh predictions with the current systems are still not pixel-accurate when projected back to the image domain. Mesh reconstruction evaluation is primarily 3D skeleton- or 3D mesh-based (measured in millimeters) and does not reflect 2D reprojection accuracy (in pixels). However, for persons close to the camera, small 3D errors result in visible

*Equal contribution
†Work done while interning at Snap Inc.

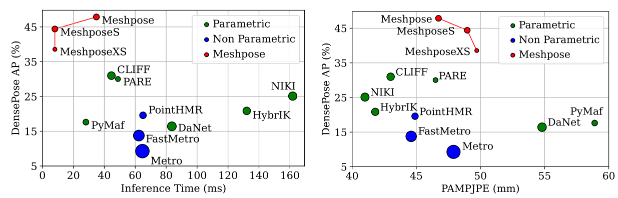


Figure 2. **Left**: Inference Time vs DensePose AP, **Right**: PA-MPJPE vs DensePose AP – for both, top-left is best and radii are proportional to the sizes of the models (MB). Our approach outperforms HMR methods on DensePose metrics by more than 50% while having close to state of the art 3D accuracy. By combining the highest FPS rate and small model size with state-of-art reprojection accuracy, our pipeline is well suited for mobile inference.

2D reprojection errors, for instance when users take selfie photos, as is the currently predominant use case for AR. If we want a 3D mesh that 'looks good' when projected to 2D the present HMR method evaluation must be complemented by 2D reprojection metrics.

Such errors are in the blindspot of 3D evaluation metrics and can break an AR experience such as virtual try-on. Errors in the 2D projection of a mesh are glaringly obvious (e.g. bags floating above the user's shoulders, coats that are too tight/too loose, misplacements around limbs etc.). DensePose has been a popular alternative to accurate warp tight garments to the user's body [1, 7, 42], however warping does not suffice for apparel (e.g. dress, coat, handbag) that protrudes from the user's body and requires proper 3D try-on.

Motivated by this observation, in this work we set out to bridge the gap between the DensePose and HMR systems. For this we revisit the DensePose task [10] and show that we can use the DensePose dataset to supervise a network that relies on a discrete, 3D vertex-based representation rather than relying on continuous UV prediction. The resulting system performs similarly to UV-based systems on the DensePose task while at the same time delivering an accurate 3D mesh. We call the resulting mesh prediction system "MeshPose" to indicate that it combines Mesh and DensePose prediction in a unified system.

To achieve this we make the following contributions:

- We introduce VertexPose, a novel layer designed to predict the 2D projections of the vertices of a low poly 3D body mesh and simultaneously regress the per-pixel DensePose UV signal directly. This is accomplished through dense heatmaps that allow us to precisely pin down the pixel coordinates of a vertex. For this we introduce two new weakly supervised losses that rely on the mesh geometry to supervise VertexPose through the DensePose dataset.
- We then introduce MeshPose to form a 3D mesh out of the localized 2D vertices. This is accomplished by regressing per-vertex a depth, visibility and amodal 3D estimate. We lift visible vertices to 3D by concatenating the 2D position with their depth, and use the amodal estimate for invisible vertices.

Our results, shown in Fig. 2 and in more detail in Table 3 show that when assessed in terms of 2D DensePose or even plain 2D pose estimation accuracy, other recent methods can use even 10× more parameters (e.g. MeshPoseXS vs Metro), or be 20× slower (e.g. MeshPoseXS vs NIKI) yet still result in substantially worse 2D reprojection metrics. At the same time our mesh reconstruction performance is comparable to most recent systems on 3DPW. Given the importance of 2D reprojection to the end-user experience in AR, we hope that our work will establish DensePose-based evaluation and training as a standard practice in future HMR works.

Our video results, provided in the Supplement, complement these findings and indicate the temporal stability of our method even when applied frame-by-frame. Our method is lightweight, simple and directly amenable to real-time inference on mobile devices, making it a prime candidate for AR applications.

2. Previous work

Our starting point for this work is the understanding that Human Mesh Reconstruction systems are typically not grounded on pixel-level evidence for vertex positions, but instead try to predict them through the reconstruction of the much more complex structure of the body mesh. We take a bottom-up approach, where we first detect visible vertices through dense 2D heatmaps and then build the mesh around them. As we explain here, this has not been an obvious approach to HMR before our work.

Parametric 3D mesh reconstruction methods such as [4, 9, 17, 21–24, 26, 29, 39, 55] methods provide rotation estimates to a forward kinematics (FK) recursion that unavoidably accumulates errors, thereby making it challenging to achieve good mesh alignment on wrists or ankles. Iterative fitting methods such as [9, 16, 24, 40, 44, 53] can mitigate this by minimizing the back-projection errors through gradient descent on the model parameters or by scaling up the required computational power [8], but both are inappropriate for real-time inference. Variants of these works have been introduced to address additional complications due to human-human occlusion or object-human occlusion [11, 15, 19, 22, 38, 45, 46, 58] and perspective distortion effects [29, 50] for in-the-wild scenes, where the problems described above become even harder.

Inverse Kinematics (IK)-based methods [14, 27, 28, 43] are a step forward when it comes to localization accuracy in that they ensure that the recovered mesh 'passes through' a set of 3D joints provided by a bottom-up system, yielding high 3D pose accuracy results. As our results show however, these methods fare poorly when it comes to 2D reprojection, since they are still constrained by the pose and shape variation of the employed parametric model.

As an alternative to IK, recurrent refinement methods such as [6, 51, 56] repeatedly estimate the positions of mesh vertices and sample features at the vertex positions to get a 'second look' at the image. Our results indicate that their 2D reprojection accuracy is limited, while their recursive projection/lookup operations make them harder to deploy for mobile AR applications since they require custom layers which are not supported in CoreML[5]/TFlite[47] and end up being computational bottlenecks when performing inference on NPU/GPU-accelerated mobile devices.

Non-parametric HMR methods such as graph-convolutional [3, 25], heatmap-based [33, 35, 54], or transformer-based models [2, 20, 30, 31] bypass parametric models and directly regress the full body mesh. In principle this can avoid the problem of error accumulation during FK, yet as our DensePose results show in Table 3 these methods still suffer when it comes to establishing accurate image reprojection. Our method bears similarity to recent methods for integral regression of per-vertex $x/y/z$ values [35, 54] in that we ground the vertex coordinates directly on image evidence, rather than regressing them through global mesh recovery systems. However we differ in that we directly connect our method to the DensePose estimation and training problems, thereby allowing us to get substantially better accuracy through direct optimization of the relevant objective. We use DensePose ground truth as weak supervision to localize our mesh vertices in 2D, and show that we can both deliver accurate 3D meshes and DensePose predictions.

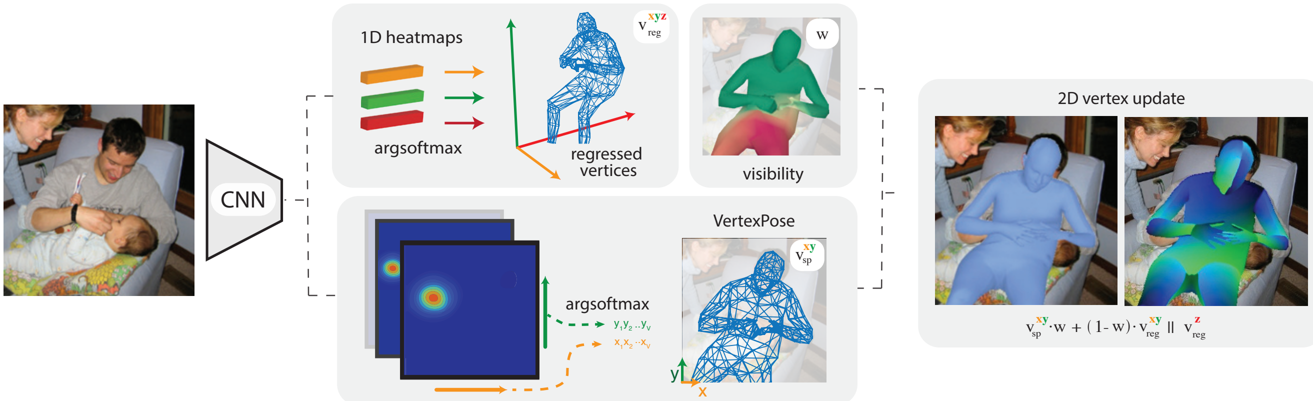


Figure 3. Meshpose Architecture: The lower VertexPose branch extracts multiple heatmaps from which, by applying the spatial argsoftmax operation, it computes precise x and y coordinates for all the vertices inside the input crop. The upper Regression branch computes the coordinates (x, y, and vertex depth z) for all vertices, along with their visibility scores w. The score w will take lower values when the corresponding vertex is either occluded or fall outside the crop area. We differentiably combine the VertexPose and regressed coordinates via w to get the final 3D mesh. We densely supervise the intermediate per-vertex heatmaps and the final output with UV, mesh and silhouette cues to end up with a low latency, image aligned, in-the-wild HMR system.

The DensePose task aims at associating every human pixel with its continuous, surface-based UV coordinates. This has been typically addressed through a dense regression task, where a CNN tries to directly recover these UV values with per-part UV regression heads, trained through a set of pixel-UV annotations. Unfortunately UV regression is of little direct value when it comes to mesh reconstruction: 2D vertex localization from DensePose requires multiple tricks (e.g. thresholding of UV distances, de-duplicating vertices, fixing left-right prediction errors for legs) and explains why this has not been adopted as a front-end processing for mesh recovery.

3. Method

In our work we recover a human body mesh from a single image in two stages as shown in Fig. 3. In the first stage, detailed in Sec. 3.1, we introduce "VertexPose", a novel layer that serves a dual purpose: predicting DensePose and localizing mesh vertices in 2D. VertexPose is designed to predict 2D heatmaps for a sparse set of body vertices that form a low-polygon approximation to a high-resolution mesh. Moreover, for each pixel, these heatmaps are integrated using barycentric combination, to compute the UV coordinates for any given pixel as DensePose. We introduce novel losses to obtain weak supervision for VertexPose heatmaps from DensePose data. We show that even though we do not have direct supervision for the 2D locations of the VertexPose vertices, we obtain DensePose accuracy comparable to UV-based DensePose systems.

In the second stage, detailed in Sec. 3.2, we lift the 2D VertexPose vertex positions to 3D, constructing a 3D mesh that accurately projects back to VertexPose vertices. We achieve this through a simple 1D integral regression task which estimates the root relative depth for all vertices in pixels.

We complement the VertexPose-based losses with 3D counterparts that allow us to exploit 3D ground-truth and pseudo ground-truth to jointly train VertexPose and its associated 3D lifted predictions.

As the upper branch of our diagram shows, we augment the VertexPose-based predictions with a per-vertex visibility estimate that is learned from weak supervision, as well as a global, image-level 3D regression of all mesh vertices. This allows us to handle invisible parts by fusing the VertexPose-based vertices with the latter prediction, which acts as a fallback.

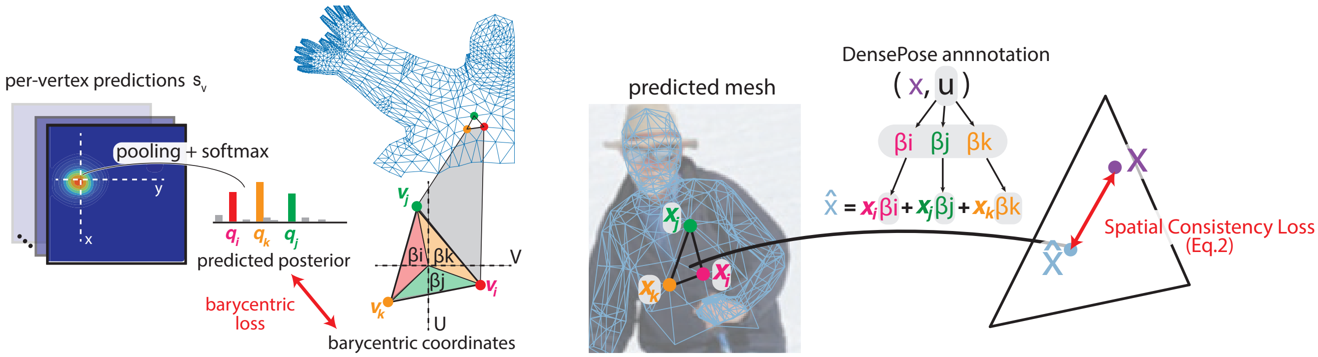
All stages rely on a single network that is trained end-to-end, but we separate their presentation and evaluation; we describe VertexPose below and MeshPose in Sec. 3.2.

3.1. VertexPose: Vertex-based DensePose

For VertexPose we draw inspiration from the success of heatmap-based systems for 2D pose estimation e.g. [52] and propose a similar layer to localize the low-poly vertices in 2D. We start by describing the operation of the VertexPose layer and then introduce new losses that allow training it from DensePose ground-truth through weak supervision.

3.1.1 VertexPose layer

The VertexPose layer consists of an $H \times W \times V$ tensor $\mathbf{S}$ where H, W are tensor height/width and V is the number of vertices. This provides for each mesh vertex the score of all 2D input positions, yielding a set of heatmaps that serve both 2D vertex localization and dense UV prediction. We further



(a) Barycentric loss (b) UV Consistency loss

Figure 4. Geometry-driven losses used to supervise VertexPose with DensePose ground-truth. Our *barycentric loss* requires that the per-pixel distribution over VertexPose matches the UV annotation's barycentrics. Our *UV consistency loss* requires that the UV annotation's barycentrics at a labelled pixel $\mathbf{x}$ should recover $\mathbf{x}$ based on a similar combination of VertexPose vertices into $\hat{\mathbf{x}}$.

denote by $s = \mathbf{S}[x, y, :]$ the $1 \times V$ set of VertexPose scores at a given position $\mathbf{x} = (x, y)$ and by $S_v = \mathbf{S}[:, :, v]$ the $H \times W$ heatmap corresponding to a given vertex v.

We localize every vertex v as the argsoftmax of S_v:

$$\mathbf{x}_v = \operatorname{argsoftmax}(S_v) = \frac{\sum_i \mathbf{x}_i \exp(\alpha S_v[\mathbf{x}_i])}{\sum_i \exp(\alpha S_v[\mathbf{x}_i])}, \quad (1)$$

where the exponentiation and normalization turns the possibly negative heatmaps into a distribution over positions and the scaling parameter α allows us to make the resulting distribution more peaked and helps with localization.

We estimate the UV value at a position $\mathbf{x}$ based on the per-pixel posterior over vertices $q_v = \frac{\exp(s_v)}{\sum_{k=1}^{V} \exp(s_k)}$. We first identify the mesh face $f = i, j, k$ whose vertices have the largest cumulative score: $f^* = \operatorname{argmax}_{f \in \mathcal{F}} \sum_{n \in f} q_n$. We then estimate the UV value as the barycentric combination of the UV values of the face vertices $\mathbf{u}_m, m \in f^*$:

$$\mathbf{u} = \sum_{m \in f^*} \beta_m \mathbf{u}_m, \text{ where } \beta_m = \frac{q_m}{\sum_{n \in f^*} q_n} \quad (2)$$

where we treat the normalized posterior over vertices that form f^* as an estimate of the pixel's barycentric coordinates.

We note the dual nature of our VertexPose layer: UV estimation is not needed at inference time for our network, but is rather used as a means to supervising our network through DensePose ground-truth. By contrast vertex localization cannot be directly supervised, but is the 2D substrate for our 3D MeshPose system.

VertexPose additionally generates a segmentation mask to accurately localize the foreground, which is a required component for the evaluation of DensePose metrics for this subsystem. This layer is omitted from the final HMR system.

3.1.2 VertexPose training

The main challenge when training the VertexPose layer is the absence of direct supervision for the 2D VertexPose vertex positions: the DensePose dataset was collected with continuous regression in mind and relied on annotating random body pixels with their associated UV values. This means that we do not have strong supervision at the level of per-vertex 2D ground-truth locations, which would allow us to directly also use the loss functions used for 2D pose estimation e.g. in [52]. We mitigate this by introducing novel losses that exploit the underlying geometric nature of VertexPose and thereby allow us to use DensePose data for weak supervision.

Barycentric Cross-Entropy Loss: This loss forces the softmax-based posterior over VertexPose vertices to approximate the barycentric coordinates on any pixel that has UV annotation, as shown in Fig. 4(a).

In particular for any pixel $\mathbf{x} = (x, y)$ that comes with a DensePose annotation with UV coordinates $\mathbf{u}$ we introduce a loss on the VertexPose scores $s = \mathbf{S}[x, y, :]$ at that pixel. We phrase the task as one of competition among the VertexPose vertices for the occupancy of the particular pixel. If a vertex v landed precisely on a given pixel we could impose at that point a standard Cross-Entropy loss using the one-hot encoding of that vertex. But this is unlikely to happen, since DensePose ground-truth was originally not sampled on specific landmark locations such as the vertices.

Instead we form our loss by interpreting the ground-truth barycentric coordinates as a discrete distribution on vertices of the ground-truth triangle f. We use this to penalize the softmax-based posterior using the general definition of the cross-entropy loss:

$$\mathcal{L}_{BL} = -\sum_v p_v \log(q_v), \text{ with } p_v = \begin{cases} \beta_v & v \in f \\ 0 & v \notin f \end{cases} \quad (3)$$

where we replace the common one-hot encoding of the correct label with a distribution on vertices, p_v, forcing the

VertexPose-based posterior $\mathbf{q}$ to align with the barycentric-based distribution $\mathbf{p}$. Our results indicate the advantage of using this geometry-inspired loss instead of a cruder, nearest neighbor assignment of annotated pixels to their nearest mesh vertex.

UV Consistency Loss: This loss forces VertexPose to place vertices so that the image coordinates of annotated UV values align with the positions where they were annotated, as shown in Fig. 4(b).

In particular we turn a pixel's DensePose annotation $(\mathbf{x}, \mathbf{u})$ into a constraint on the VertexPose heatmaps $S_v = \mathbf{S}[:, :, v], v \in (i, j, k)$ of the three vertices (i, j, k) used to compute the barycentric coordinates $(\beta_i, \beta_j, \beta_k)$ of $\mathbf{u}$. These three barycentric coordinates allow us to localize the pixel's corresponding point on the 3D surface as a convex combination of these three vertices. When projected to the image this relationship should still roughly hold, modulo depth-based perspective distortion effects, which we consider negligible within a triangle. Our loss enforces this: when combining the estimated 2D positions of the three vertices $\mathbf{x}_v = \text{argsoftmax}(S_v), v \in \{i, j, k\}$, with barycentric weights β_v, we should be able to recover the position of $\mathbf{x}$:

$$L_{\text{consistency}} = \|\mathbf{x} - \hat{\mathbf{x}}\|, \text{ where } \hat{\mathbf{x}} = \sum_{v \in \{i,j,k\}} \beta_v \mathbf{x}_v \quad (4)$$

This forces the heatmap S_v to properly localize $\mathbf{x}_v$. without direct supervision for vertex v.

Both of these losses are efficient to evaluate and as our results in Sec. 4 show, they add up to the training of a VertexPose system that even outperforms the UV-based DensePose baseline when trained with identical data and experimental settings. Still, we consider the competition with UV-based DensePose systems to be of secondary importance compared to being able to directly predict a mesh based on the subsequent lifting of the estimated vertices to 3D, as described in Sec. 3.2.

3.2. MeshPose: Lifting VertexPose to 3D

Having outlined our method to localize mesh vertices in 2D we now turn to converting them into a 3D mesh. As shown in Fig. 3, our method consists of retaining the image localization information of VertexPose where available and filling in the remaining information by values regressed by a separate network branch.

Inspired by [35, 43] we take the backbone CNN's last tensor, average pool it and transform the result through 1D convolutional layers to regress a $4 \times 64 \times V$ tensor, where V is the number of the low-poly vertices, the four channels correspond to X, Y, Z values and a per-vertex visibility label w and 64 are the number of bins used for argsoftmax voting. In particular for every regressed vertex V_{reg}^{XYZ} its X, Y, Z values are obtained separately per dimension by applying 1D argsoftmax while the visibility w is obtained by mean pooling followed by a sigmoid unit. We detail how we supervise those terms below.

3.2.1 Visibility prediction

The visibility label dictates on a per-vertex level whether we should rely on the VertexPose-based 2D position, V_{sp}^{XY} or fall back to the V_{reg}^{XY} value regressed at this stage. This allows us to accommodate occluded areas, or tight crops that omit part of the human body, as is regularly the case for selfie images. The 2D location of a MeshPose vertex is their visibility-weighted average: $V_{mp}^{XY} = V_{sp}^{XY} w + V_{reg}^{XY} (1-w)$. This differentiable expression allows us to estimate visibility through end-to-end back-propagation, but we also use two additional methods for visibility supervision.

Firstly we estimate partial vertex visibility based on the available ground-truth: for any $(\mathbf{x}, \mathbf{u})$ annotation pair contained in the DensePose dataset, we declare as visible all three vertices that lie on the mesh triangle containing $\mathbf{u}$. We also declare as non-visible every vertex where the mesh supervision (obtained from [37]) is outside the image crop. For such vertices we can supervise visibility based on a standard binary cross-entropy loss.

Secondly, we also supervise visibility at the mesh level. For this we use differentiable rendering with the per-vertex texture set to equal the predicted visibility label. This produces a soft visibility mask, shown also in Fig. 3 as a heatmap, which indicates the image area that is covered by the person's body. This can be supervised at the region level based on the DensePose dataset's instance segmentation masks, using a mix of an ℓ_2 loss with the integral boundary loss introduced in [18]. These two sources of visibility supervision gave substantial improvements as also shown by our results.

3.2.2 Depth regression

We adopt a weak perspective camera model as in [35, 54] and consider that each vertex lies on a ray that crosses the image plane at the VertexPose-based 2D position. We thereby limit 3D lifting to the task of estimating the vertex depth on that ray. Rather than directly regress the depth of a vertex, we predict its depth relative to the 'root' of the mesh (sternum). The latter is predicted by a separate RootNet network [36], leaving to our network the task of relative depth estimation. We estimate depth in pixel units, based on the same rigid transform used to associate the metric joint (x,y) positions with their 2D pixel counterparts.

3.2.3 MeshPose output

The MeshPose 3D prediction concatenates the visibility-weighted 2D location with the depth prediction:

$$V_{mp}^{XYZ} = (V_{sp}^{XY} w + V_{reg}^{XY} (1-w)) || V_{reg}^{Z}) \quad (5)$$

All terms are differentiable, allowing us to train our network end-to-end based on 3D mesh supervision.

We note that we can optionally also transform the resulting low-poly mesh into a high-poly counterpart through an MLP-based upsampling as in [30]. We have trained such an MLP and used it both for mesh visualizations and performance evaluations. Even though it primarily serves as a "visual embelishment" of the low-poly prediction, it also provides some form of regularization when trained with noised low-poly inputs.

Finally, if a parametric representation is needed for an application the MeshPose prediction lends itself easily to Inverse Kinematics-based processing [27] by using landmarker-based 3D joint estimates. Our implementation of HybridIK-type decoding yields virtually identical 3D metrics, but comes at the cost of a drop in DensePose accuracy, as expected, hence we omit it from evaluations.

3.2.4 3D supervision

In order to supervise the 3D coordinates of the low poly mesh we use motion capture ground truth (GT) meshes, weak supervision for the in-the-wild COCO dataset (pseudo-GT), and the losses described below.

Vertex Localization, Edge and Normal Loss: The position of our 3D vertices can be directly compared to the (pseudo) GT in terms of an L2 loss ('localization loss'), while we can also penalize the distortion of the edge lengths between two adjacent vertices ('edge loss'). We note that the second loss does not necessarily guarantee good alignment to the image, but ensures we do not arbitrarily stretch or shrink the mesh to reduce other losses. To further reduce mesh curvature artifacts we use a third, ('normal cosine loss') that penalizes the deviation of predicted vertex normals.

Joint Localization Loss: The last form of supervision relies on standard 2D or 3D joint GT that is more readily available through image annotations or motion capture, respectively. The position of each joint is estimated as a weighted average of a subset of nearby mesh vertices, implemented as a precomputed linear regression (landmarker). We compare the predicted joints to the GT using its full 3D coordinates or their projections on the image, based on whether we have 3D or 2D supervision. Even though only a sparse subset of vertices contribute to the prediction of any joint, the edge loss described above helps diffuse the supervision to the remainder of the mesh.

4. Results

We start by describing experimental settings, and then proceed with quantitative and qualitative evaluation. *Due to lack of space we provide additional ablations and experimental results in the Supplemental Material, including results on videos.*

4.1. Datasets

We use the manual DensePose annotations provided in [10] on the **MS-COCO** [32] dataset for training the Vertex-Pose system. For 3D joint supervision we use three datasets: (i) The **Human3.6M** [13]: MoCap dataset and follow the protocol of [24] (subjects (S1, S5, S6, S7, S8) for training) (ii) The **MPI-INF-3DHP** [34]: Multi-view dataset, following the train split of [24]. (iii) The **3DPW** [48] in-the-wild outdoor benchmark for 3D pose and shape estimation containing 3D annotations from IMU devices. Furthermore, we augment the MS-COCO dataset with the 3D mesh pseudo GT annotations of [37] for vertex-level mesh supervision.

4.2. Evaluation Metrics

We adopt the evaluation framework outlined in [10] for the DensePose task, where we report on two key metrics: **AP** (Average Precision) and **AR** (Average Recall). These metrics quantify the accuracy of the dense correspondences from UV coordinate predictions on images. For mesh recovery methods this can be interpreted as measuring mesh alignment accuracy after projection. We measure the correspondence accuracy by rendering UV coordinates of the visible mesh surface. In addition to these 2D metrics, we evaluate the accuracy of 2D COCO keypoints prediction, quantified through Average Precision and Recall. This evaluation is conducted both across all instances (**AP-All** and **AR-All**), as well as only on instances where at least 80% of the keypoints are visible (**AP-80%** and **AR-80%**). All 2D metrics are evaluated on DensePose-COCO, a subset of COCO [32] introduced in [10].

For 3D pose evaluation we employ a landmarker on top of the high poly 3D mesh to compute the 14 LSP joints for the evaluation on 3DPW dataset [17, 24]. Then we compute the Euclidean distances (in millimeter (mm)) of 3D points between the predictions and GT as described by the following metrics: (i) **MPJPE** (Mean Per Joint Position Error) first aligns the predicted and GT 3D joints at the 3D position of the pelvis, evaluating the predicted pose by taking into account the global rotation. (ii) **PA-MPJPE** (Procrustes-Aligned Mean Per Joint Position Error, or reconstruction error) performs Procrustes alignment before computing MPJPE, eliminating any error due to wrong global scale and global rotation. (iii) **PVE** (Per Vertex Error) does the same alignment as MPJPE and then calculates the distances of vertices of human mesh, evaluating also the mesh shape additionally to the 3D skeleton pose.

4.3. VertexPose evaluation

We start by examining the impact of training with the vertex-based (VertexPose) approach compared to the UV-regression based (DensePose) representations for the DensePose task. To keep the comparison apples-to-apples in Table 1a we compare experiments with identical backbones and training settings, where we closely follow [10] for designing the DensePose baseline. We observe that the VertexPose-based results compare favorably to their DensePose-based counterparts, confirming the validity of the proposed approach.

In Table 1b we analyze the impact of the barycentric interpolation strategy used to predict UV in Eq. 2. We compare it to simpler baselines of using the UV of the strongest vertex at any pixel ('Nearest') or doing argsoftmax over all vertices rather than those of the strongest triangle ('Global Average'). The results indicate the merit of the smooth transition between vertices secured by barycentric interpolation.

We consider improvements in the 2D DensePose task to be of secondary importance compared to improving the 3D HMR's DensePose performance, hence keep the remaining results focused on MeshPose.

	AP	**$\mathbf{AP}_{50}$**	**AR**	**$\mathbf{AR}_{50}$**
DP - MBNet	53.54	91.66	64.09	95.68
SP - MBNet	54.08	93.01	64.46	96.03
DP - ResNet	57.31	93.06	67.51	96.30
SP - ResNet	58.87	93.05	68.73	96.26
DP - HRNet32	61.12	94.84	70.53	97.33
SP - HRNet32	61.24	94.63	70.52	97.10
DP - HRNet48	62.74	95.04	71.95	97.50
SP - HRNet48	63.32	95.12	72.14	97.73

(a) VertexPose vs DensePose.

	AP	**$\mathbf{AP}_{50}$**	**AR**	**$\mathbf{AR}_{50}$**
Barycentric	61.24	94.63	70.52	97.10
Closest	59.69	95.10	68.80	97.59
Global Average	33.35	89.58	43.23	94.38

(b) UV aggregation strategy

Table 1. Analysis of VertexPose performance on the DensePose-COCO dataset. We evaluate the impact of the backbone choice and the UV decoding strategy.

4.4. MeshPose evaluation

In Table 2 we start by ablating the impact of the types of supervision used for our full MeshPose system. Starting from only 3D losses (vertex localization, edge and normal loss - $\mathcal{L}_{3D}$) on row 1, we show the impact of adding 2D losses (barycentric and uv consistency loss - $\mathcal{L}_{2D}$) and visibility supervision (partial visibility and rendering loss - $\mathcal{L}_{W}$). We show that adding 2D losses leads to a moderate drop in 3D accuracy but boosts DensePose reprojection metrics, while adding visibility supervision helps improve both tasks at the same time - effectively helping them coexist.

Turning to comparisons with HMR methods, in (Table 3) we extensively compare our approach with 10 other SOTA architectures in terms of efficiency (measured in #Parameters and FPS) and accuracy. We report the performance on 3 tasks, (i) 2D DensePose-COCO Keypoints, (ii) DensePose and (iii) 3D alignment on 3DPW and Human 3.6M.

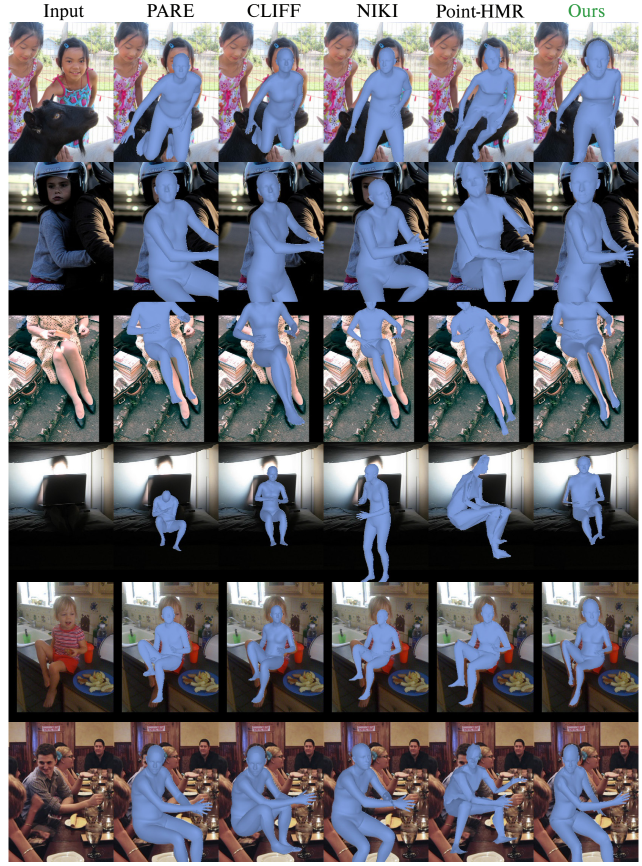


Figure 5. Qualitative comparison on COCO against 4 state-of-the-art mesh reconstruction systems. MeshPose is robust to severe occlusions, partial body cropping and body shapes.

Losses			3DPW			COCO-DensePose		
$\mathcal{L}_{3D}$	$\mathcal{L}_{2D}$	$\mathcal{L}_{W}$	MPJPE	PA-MPJPE	PVE	AP	AR	IoU
✓			**76.52**	**45.31**	**91.88**	38.22	49.07	56.56
✓	✓		81.37	50.40	101.68	44.09	53.31	57.64
✓	✓	✓	77.10	46.54	94.78	**45.51**	**55.18**	**60.24**

Table 2. Ablation table evaluated in terms of 3D metrics (3DPW) and 2D reprojection accuracy (COCO-DensePose).

MeshPose is by far the most efficient approach for achieving real-time prediction while getting the best 2D reprojection performance. This large improvement comes at the cost of a small impact on 3D metrics.

Our method achieves competitive performance against other parametric and non-parametric approaches, while largely outperforming them on the DensePose task. More specifically, MeshPose achieves much better DensePose metrics compared against CLIFF and NIKI methods, which have the best scores for the HMR task.

Fig. 5 depicts some examples comparing the proposed MeshPose against the PARE, CLIFF, NIKI and Point-HMR methods. We can see that MeshPose produces meshes that, when projected on the image, align much better than compet-

	Method	Efficiency		2D COCO KeyPoints ↑				DensePose ↑		3DPW ↓			Human 3.6M ↓	
		#Params ↓	FPS ↑	AP-All	AR-All	AP-80%	AR-80%	AP	AR	MPJPE	PAMPJPE	PVE	MPJPE	PAMPJPE
param	DaNet [57]	102.25	11.95	19.80	36.20	52.60	65.20	16.42	29.24	85.50	54.80	110.80	54.60	42.90
	HybrIK [27]	75.69	7.57	10.60	24.90	31.70	47.50	20.85	34.44	71.60	41.80	82.30	**47.00**	**29.80**
	PARE [22]	32.86	20.48	33.20	48.10	56.90	67.20	30.02	41.54	74.50	46.50	88.60	-	-
	CLIFF [29]	78.89	22.42	33.20	49.30	61.40	72.50	30.99	41.83	**69.00**	43.00	**81.20**	47.10	32.70
	PyMaf [56]	45.18	35.57	23.00	40.40	58.00	70.00	17.62	30.69	92.80	58.90	110.10	57.70	40.50
	NIKI [28]	92.17	6.18	21.30	37.20	48.50	61.50	25.11	37.39	71.70	**41.00**	86.90	-	-
n-param	Metro [30]	243.07	15.47	9.50	22.40	30.40	45.10	9.28	21.16	77.10	47.90	88.20	54.40	34.50
	Graphormer [31]	226.21	14.42	12.00	25.70	35.90	49.50	13.57	26.20	74.70	45.60	87.70	51.20	34.50
	FastMetro [2]	153.74	16.04	13.60	28.00	39.30	53.20	13.76	26.21	73.50	44.60	84.10	52.20	33.70
	PointHMR [20]	59.09	15.38	17.70	32.70	44.40	57.30	19.58	32.18	73.90	44.90	85.50	48.30	32.90
ours	MeshPose (HRNet32 [49])	46.29	28.66	**47.30**	**61.30**	**71.20**	**79.60**	**47.87**	**57.62**	76.08	46.73	92.70	50.76	35.37
	MeshPoseS (ResNet50 [12])	45.37	**124.28**	43.80	58.10	67.00	76.50	44.41	54.49	80.03	48.97	97.96	56.33	37.64
	MeshPoseXS (MBNet140 [41])	**21.25**	124.21	40.60	55.20	63.60	73.90	38.56	49.20	79.15	49.71	96.49	58.40	41.63

Table 3. Evaluation of network efficiency, 2D accuracy in COCO-DensePose and 3D errors in the 3DPW and Human3.6M datasets. The variants of our system achieve superior performance in 2D metrics (2D Keypoints, Densepose) when compared to other methods, while they achieve comparable 3D accuracy. At the same time they are substantially more efficient in terms of FPS and # of parameters.

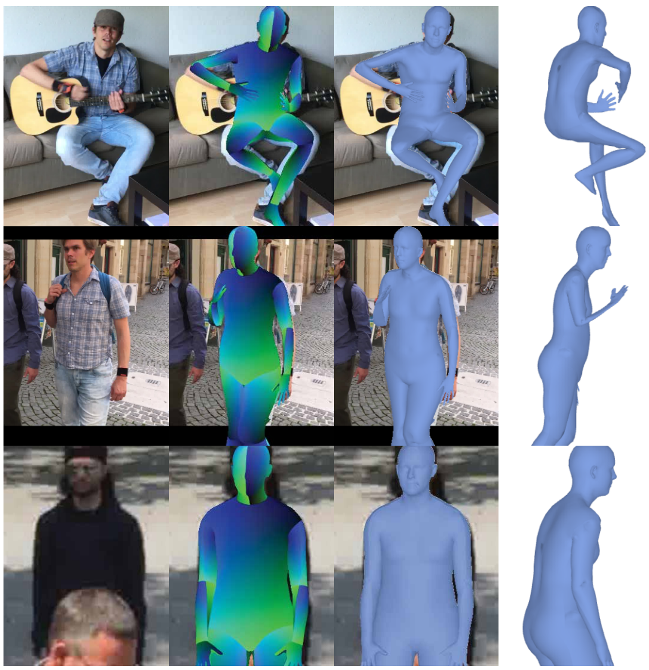


Figure 6. Qualitative results on 3DPW on front and side views. Our method shows strong 2D alignment with accurate 3D mesh prediction.

ing methods. Other methods fail e.g. when reconstructing children, when a large part of the body is occluded and have inferior alignment around the limbs. In Figure 6, we further demonstrate the performance of our system in terms of mesh reconstruction by including side views of our meshes predicted on 3DPW images.

4.5. Limitations

As shown on Figure 7, the two main failure modes of our system are hand/arm flattening artifact and imperfect alignment for perspectively distorted inputs. The flattening artifact primarily arises due to the mesh upsampler's lack of training on examples with articulated hands. The imperfect perspective alignment is caused by our system's reliance on the assumption of weak perspective camera model. Still, the robustness of our method is comparable to DensePose, hence we do not have catastrophic failures (e.g wrong torso pose) in the presence of heavy occlusions.

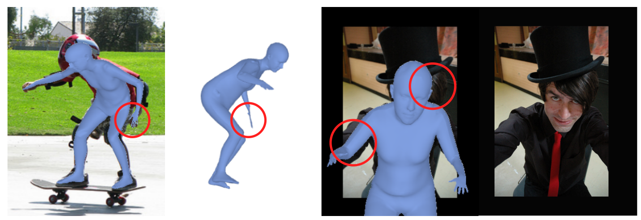


Figure 7. Typical failure cases of MeshPose include a hand flattening artifact and imperfect image alignment in the presence of perspective distortion.

5. Conclusion

In this work, we started by observing the limited 2D reprojection accuracy of current HMR systems, which limits their applicability to augmented reality applications - e.g. virtual try-on for garments and accessories. To address this we have introduced MeshPose, a system that bridges the DensePose and HMR problems and substantially improves the image reprojection accuracy of HMR, while retaining accurate 3D pose. Beyond improved accuracy, our approach relies on a lightweight and simple architecture that consists of only standard neural networks layers. This makes our approach a natural fit for AR applications requiring real-time mobile inference.

References

[1] Badour AlBahar, Jingwan Lu, Jimei Yang, Zhixin Shu, Eli Shechtman, and Jia-Bin Huang. Pose with Style: Detail-preserving pose-guided image synthesis with conditional stylegan. *ACM Transactions on Graphics*, 2021. 1

[2] Junhyeong Cho, Kim Youwang, and Tae-Hyun Oh. Cross-attention of disentangled modalities for 3d human mesh recovery with transformers. In *ECCV*, 2022. 2, 8

[3] Hongsuk Choi, Gyeongsik Moon, and Kyoung Mu Lee. Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose. In *ECCV*, 2020. 2

[4] Hongsuk Choi, Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Beyond static features for temporally consistent 3d human pose and shape from a video. In *CVPR*, 2021. 2

[5] Core ML. https://developer.apple.com/documentation/coreml. 2

[6] Enric Corona, Gerard Pons-Moll, Guillem Alenyà, and Francesc Moreno-Noguer. Learned vertex descent: A new direction for 3d human model fitting. *CVPR*, 2022. 2

[7] Chenghu Du, Shuqing Liu, Shengwu Xiong, et al. Greatness in simplicity: Unified self-cycle consistency for parser-free virtual try-on. *NeurIPS*, 36, 2024. 1

[8] Shubham Goel, Georgios Pavlakos, Jathushan Rajasegaran, Angjoo Kanazawa*, and Jitendra Malik*. Humans in 4D: Reconstructing and tracking humans with transformers. In *ICCV*, 2023. 2

[9] Riza Alp Guler and Iasonas Kokkinos. Holopose: Holistic 3d human reconstruction in-the-wild. In *CVPR*, 2019. 2

[10] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *CVPR*, 2018. 2, 6, 7

[11] Mohamed Hassan, Vasileios Choutas, Dimitrios Tzionas, and Michael J Black. Resolving 3d human pose ambiguities with 3d scene constraints. In *ICCV*, 2019. 2

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 8

[13] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE TPAMI*, 36(7):1325–1339, 2013. 6

[14] Umar Iqbal, Kevin Xie, Yunrong Guo, Jan Kautz, and Pavlo Molchanov. Kama: 3d keypoint aware body mesh articulation. In *International Conference on 3D Vision*, 2021. 2

[15] Wen Jiang, Nikos Kolotouros, Georgios Pavlakos, Xiaowei Zhou, and Kostas Daniilidis. Coherent reconstruction of multiple humans from a single image. In *CVPR*, 2020. 2

[16] Hanbyul Joo, Natalia Neverova, and Andrea Vedaldi. Exemplar fine-tuning for 3d human pose fitting towards in-the-wild 3d human pose estimation. In *3DV*, 2020. 2

[17] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018. 2, 6

[18] Hoel Kervadec, Jihene Bouchtiba, Christian Desrosiers, Eric Granger, Jose Dolz, and Ismail Ben Ayed. Boundary loss for highly unbalanced segmentation. In *Proceedings of The 2nd International Conference on Medical Imaging with Deep Learning*, pages 285–296. PMLR, 2019. 5

[19] Rawal Khirodkar, Shashank Tripathi, and Kris Kitani. Occluded human mesh recovery. In *CVPR*, 2022. 2

[20] Jeonghwan Kim, Mi-Gyeong Gwon, Hyunwoo Park, Hyukmin Kwon, Gi-Mun Um, and Wonjun Kim. Sampling is Matter: Point-guided 3d human mesh reconstruction. In *CVPR*, 2023. 2, 8

[21] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *CVPR*, 2020. 2

[22] Muhammed Kocabas, Chun-Hao P. Huang, Otmar Hilliges, and Michael J. Black. PARE: Part attention regressor for 3D human body estimation. In *ICCV*, 2021. 2, 8

[23] Muhammed Kocabas, Chun-Hao P Huang, Joachim Tesch, Lea Müller, Otmar Hilliges, and Michael J Black. Spec: Seeing people in the wild with an estimated camera. In *ICCV*, 2021.

[24] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *ICCV*, 2019. 2, 6

[25] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *CVPR*, 2019. 2

[26] Nikos Kolotouros, Georgios Pavlakos, Dinesh Jayaraman, and Kostas Daniilidis. Probabilistic modeling for human mesh recovery. In *ICCV*, 2021. 2

[27] Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *CVPR*, 2021. 2, 6, 8

[28] Jiefeng Li, Siyuan Bian, Qi Liu, Jiasheng Tang, Fan Wang, and Cewu Lu. NIKI: Neural inverse kinematics with invertible neural networks for 3d human pose and shape estimation. In *CVPR*, 2023. 2, 8

[29] Zhihao Li, Jianzhuang Liu, Zhensong Zhang, Songcen Xu, and Youliang Yan. Cliff: Carrying location information in full frames into human pose and shape estimation. In *ECCV*, 2022. 2, 8

[30] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. In *CVPR*, 2021. 2, 6, 8

[31] Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *ICCV*, 2021. 2, 8

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6

[33] Xiaoxuan Ma, Jiajun Su, Chunyu Wang, Wentao Zhu, and Yizhou Wang. 3d human mesh estimation from virtual markers. In *CVPR*, 2023. 2

[34] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *3DV*, 2017. 6

[35] Gyeongsik Moon and Kyoung Mu Lee. I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and

mesh estimation from a single rgb image. In *ECCV*, 2020. 2, 5

[36] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image. In *ICCV*, 2019. 5

[37] Gyeongsik Moon, Hongsuk Choi, and Kyoung Mu Lee. Neuralannot: Neural annotator for 3d human mesh training sets. In *CVPRW*, 2022. 5, 6

[38] Lea Muller, Ahmed AA Osman, Siyu Tang, Chun-Hao P Huang, and Michael J Black. On self-contact and human pose. In *CVPR*, 2021. 2

[39] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *CVPR*, 2018. 2

[40] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *CVPR*, 2019. 2

[41] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 8

[42] Kripasindhu Sarkar, Vladislav Golyanik, Lingjie Liu, and Christian Theobalt. Style and pose control for image synthesis of humans from a single monocular view, 2021. 1

[43] Karthik Shetty, Annette Birkhold, Srikrishna Jaganathan, Norbert Strobel, Markus Kowarschik, Andreas Maier, and Bernhard Egger. Pliks: A pseudo-linear inverse kinematic solver for 3d human body estimation. In *CVPR*, 2023. 2, 5

[44] Jie Song, Xu Chen, and Otmar Hilliges. Human body model fitting by learned gradient descent. In *ECCV*, 2020. 2

[45] Yu Sun, Qian Bao, Wu Liu, Yili Fu, Michael J Black, and Tao Mei. Monocular, one-stage, regression of multiple 3d people. In *ICCV*, 2021. 2

[46] Yu Sun, Wu Liu, Qian Bao, Yili Fu, Tao Mei, and Michael J Black. Putting people in their place: Monocular regression of 3d people in depth. In *CVPR*, 2022. 2

[47] Tensorflow Lite. https://www.tensorflow.org/lite/guide. 2

[48] Timo von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *ECCV*, 2018. 6

[49] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE TPAMI*, 43(10): 3349–3364, 2020. 8

[50] Wenjia Wang, Yongtao Ge, Haiyi Mei, Zhongang Cai, Qingping Sun, Yanjun Wang, Chunhua Shen, Lei Yang, and Taku Komura. Zolly: Zoom focal length correctly for perspective-distorted human mesh reconstruction. In *CVPR*, 2023. 2

[51] Yufu Wang and Kostas Daniilidis. Refit: Recurrent fitting network for 3d human recovery. In *ICCV*, 2023. 2

[52] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *ECCV*, 2018. 3, 4

[53] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Ghum & ghuml: Generative 3d human shape and articulated pose models. In *CVPR*, 2020. 2

[54] Chun-Han Yao, Jimei Yang, Duygu Ceylan, Yi Zhou, Yang Zhou, and Ming-Hsuan Yang. Learning visibility for robust dense human body estimation. In *ECCV*, 2022. 2, 5

[55] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Weakly supervised 3d human pose and shape reconstruction with normalizing flows. In *ECCV*, 2020. 2

[56] Hongwen Zhang, Yating Tian, Xinchi Zhou, Wanli Ouyang, Yebin Liu, Limin Wang, and Zhenan Sun. Pymaf: 3d human pose and shape regression with pyramidal mesh alignment feedback loop. In *ICCV*, 2021. 2, 8

[57] Hongwen Zhang, Jie Cao, Guo Lu, Wanli Ouyang, and Zhenan Sun. Learning 3d human shape and pose from dense body parts. *IEEE TPAMI*, 44(5):2610–2627, 2022. 8

[58] Tianshu Zhang, Buzhen Huang, and Yangang Wang. Object-occluded human shape and pose estimation from a single color image. In *CVPR*, 2020. 2